# ИНФОРМАТИКА И УПРАВЛЕНИЕ В ТЕХНИЧЕСКИХ И СОЦИАЛЬНЫХ СИСТЕМАХ

**УДК 004.054, 004.85**

**Н. В. Волжанкин, Н. В. Злобина, Н. Е. Пособилов**

## РАЗРАБОТКА И ВЫБОР СРЕДСТВ И МЕТОДИК ДЛЯ ТЕСТИРОВАНИЯ КОММЕРЧЕСКИХ СИСТЕМ, ПОСТРОЕННЫХ НА МАШИННОМ ОБУЧЕНИИ

Нижегородский государственный технический университет им. Р. Е. Алексеева

Анализируются современные алгоритмы машинного обучения и их применение в практических задачах и проектах. Выделяется стадия прототипирования систем с машинным обучением, разработка модели, подходы к тестированию и выявлению ошибок и слабостей проектируемой системы. Рассматриваются современные технологии, позволяющие организовать тестирование на ранних этапах проектирования. Результатом является методика, позволяющая избежать ошибок в системе на самом раннем этапе разработки, что дает возможность минимизировать издержки на исправление дефектов на более поздних этапах.

*Ключевые слова: машинное обучение, алгоритмы, тестирование, методики, проектирование систем, коммерческие системы.*

Все больше и больше компаний, особенно крупных, развивают и внедряют программное обеспечение на основе машинного обучения. Современные технологии и вычислительные мощности позволяют обрабатывать огромные объемы данных, в современном языке был введен такой термин, как Big Data, что позволит крупным компаниям проводить более глубокую аналитику на основе имеющихся баз данных, сформированных за многие годы.

Машинное обучение – это использование некоторого алгоритма обработки и анализа данных для дальнейшего принятия решения или предсказания поведения. Такой подход позволяет решить главную проблем создания таких систем программистами – установить множественные связи между исходными данными и конечным выводом. Представим, что у нас есть база данных, где каждому клиенту соответствует некий набор параметров. Этих параметров может, быть много, более тысячи, характеризующие клиента со всех сторон. Это личная информация, количество обращений, время обращения, география обращения клиента, информация для связи и т.д. Чтобы программист смог сделать программу, учитывающую все эти параметры и делать некий вывод, потребовалось бы несколько лет и необязательно, что все пройдет гладко и мы правильно интерпретируем начальные данные. Поэтому формирование этих связей необходимо переложить на машину. На основе некой обучающей выборки машина постепенно подстраивает связи между вводом и выводом, которые мы заранее определили без участия программы. Когда мы полностью обучаем машину на имеющихся у нас данных – она сможет предсказывать вывод на новых данных от будущих клиентов.

Машинное обучение придумали создатели искусственного интеллекта, и сейчас алгоритмические подходы включают обучение дереву принятия решений, индуктивное логическое программирование, кластеризацию, обучение с подкреплением, Байесовы сети и т.д.

Нет никаких сомнений в том, что машинное обучение находится на вершине кривой

зрелости технологий, что вызывает очень важные изменения в бизнес-процессах многих компаний.

Речь идет не столь о крупных гигантах, которые вкладывают огромные бюджеты в исследовательские проекты, вроде Google и Microsoft. В реальном мире все из списка Fortune 500 увеличивают свою капитализацию с помощью систем, построенных на машинном обучении. Это позволяет:

1. **Сделать пользовательский контент ценным ресурсом**. Если взять контент пять лет назад и современный из некоторых источников – он совершенно низкого качества. Опечатки, ложная информация и прочее факторы, превращающие в совершенно нечитабельной для клиента. Благодаря же машинному обучению мы сможем отфильтровать его, проранжировать самый ценный для нас и отбросить ненужный. Самый простой пример – определение спама, что уже сейчас применяется во многих сервисах.

2. **Найти продукт быстрее.** Активно эту тенденцию подхватили все поисковые компании. Даже компания Google поставила во главу поискового подразделения специалиста по машинному обучению. Машинное обучение позволяет более глубоко анализировать запрос, который указал пользователь, в купе с его историей поиска, интересов и потребностей, и способен учитывать все факторы, что повышает релевантность конечно результата для пользователя.

3. **Привлечь интерес покупателей.** Сейчас многие обратили внимание, что все реже в интернет-магазинах мы можем встретить форму «свяжитесь с нами». Компании тратят большие издержки на маршрутизацию пользователя, машинное обучение, благодаря обучающемуся алгоритму можно проанализировать суть запроса и направить пользователя в нужное место.

4. **Понять поведение клиента.** Огромная область, особенно с точки зрения маркетинга. Например, работа с эмоциональным окрасом высказываний, многие решения принимались во многом, благодаря подобному поведению. Можно в режиме реального времени отслеживать пожелания пользователей, чтобы предугадать изменения, которые нужно внести для обхвата большей аудитории. Каналом для отслеживания мнений сейчас предостаточно, в том числе и социальные сети.

На основе всего изложенного можно с уверенностью заявить, что систем, основанных на машинном обучении, будет появляться все больше и больше. Однако построение таких систем – задача, которая таит под собой множество подводных камней. На проектирование и построение подобных программы могут тратится огромные бюджеты и ошибки, допущенные на всех этапах производства, могут свести выгоду от введения в эксплуатацию подобных решений к нулю, если не обернутся убытками для компаний. Именно поэтому важной частью жизненного цикла таких систем является тестирования на всех этапах.

Области знаний, необходимые для успешной разработки систем с машинным обучением, представлены на рис. 1.


**Рис. 1. Области знаний, необходимых для работы с машинным обучением**

Тестирование информационных систем является ключевым аспектом по снижению затрат на издержки при вводе ПО в промышленную эксплуатацию. Для эффективного тестирования необходимо разработать общую модель процесса, используя самые современные средства и подходы, чтобы обеспечить максимальную эффективность.

Для реализации автоматического тестирования алгоритма необходима реализация экспериментов, выявляющих его характеристики. Для того, чтобы алгоритм был востребован, его эффективность должна быть подтверждена экспериментами на реальных данных. Эксперименты должны быть легко воспроизводимыми, чтобы любой ранее полученный результат можно было перепроверить, и стандартизованными, чтобы результаты, полученные разными людьми, были сопоставимы. В машинном обучении точность воспроизводимости достичь довольно сложно, так как это требует соблюдения ряда условий.

1. *Идентичность реализации алгоритма*. Поскольку один и тот алгоритм можно представить в разном виде, в том числе он может быть реализован на разных языках программирования либо же просто формально описан. К тому же, в случае коммерческих алгоритмов, код реализации недоступен, поскольку несет коммерческую тайну.

2. *Идентичность методики тестирования*. Описания методики кросс-валидации часто поверхностны и не содержат таких важных деталей, как стратификация выборки по классам или признакам. Более того, точное воспроизведение результатов эксперимента невозможно без знания множества разбиений выборки на обучение и контроль, которое, как правило, генерируется случайным образом и не запоминается.

3. *Идентичность исходных данных*. Хотя данные, как правило, берутся из одного источника (чаще всего из репозитория UCI), детали предварительной обработки (нормировка, заполнение пропусков, и т.д.) тоже часто опускаются.

Различия в конечных показателях результатов экспериментов могут существенно зависеть от проблем, описанных ранее. Все это может повлиять на наш конечный выбор, хотя данные показатели совершенно необъективны. Если полученные результаты расходятся, то причиной может быть различие и в реализации алгоритмов, и в методике тестирования, и в данных.

Для решения данных проблем были попытки создания систем для автоматизации экспериментов в машинном обучении. Они наталкивались на технологические и организационные трудности, однако развитие Web-технологий привело к созданию нескольких общедоступных систем: TunedIt, MLComp. Подобные системы в своей основе имеют пополняемый репозиторий задач и алгоритмов. Каждая из них использует для тестирования алгоритмов методику кросс-валидации, которая заключается в множественном разбиении данных, используемых при обучении алгоритма (пары входные данные – результат), на категории «обучение» и «контроль», что позволит вычислять показатели качества решаемой задачи (вычислять среднюю ошибку на контроле). Результаты экспериментов выдаются через веб-интерфейс. Однако каждая из систем имеет свои отличия, например в том, как исполняются алгоритмы, какие типы допускаются, где они хранятся, как происходит тестирование, какие характеристики измеряются и что выдается конечному пользователю.

**Система TunedIt**. Система состоит из трех блоков – «базы знаний», «репозитория задач и алгоритмов» и «тестировщика» (*TunedTester*). *TunedTester* позволяет запускать алгоритмы и выгружать результаты тестирования на сервер системы. Этот компонент работает локально, поэтому пользователь должен загрузить и установить на свой компьютер программу с сайта www.tunedit.org. Все необходимые для тестирования задачи и алгоритмы загружаются на компьютер пользователями автоматически при запуске тестирования. Алгоритмы, используемые *TunedTester'*ом должны быть написаны на языке JAVA. Компонент работает с алгоритмами из библиотек WEKA, Rselib и Debellor. Для добавления алгоритма надо встроить его в одну из этих трех библиотек. В системе есть стандартная

процедура тестирования – разбиение задач на обучение и контроль (случайное, 70/30). Качество алгоритма оценивается как доля ошибок на контроле. Пользователь может дополнить систему собственной процедурой тестирования (*evalution procedure)*, использующей произвольные разбиения задачи и выдающей на выходе любую другую скалярную величину. База знаний предоставляет интерфейс для сравнения результатов различных алгоритмов. Сравнение возможно только в контексте одной задачи и одной процедуры тестирования. Для каждого алгоритма указывается количество запусков процедуры тестирования, среднее значение и стандартное отклонение данной оценки качества на данной задаче. Для стандартной процедуры тестирования оценкой является доля ошибок на контроле.

Ограничения:

1. Ограничение на язык программирования (JAVA).
2. Отсутствуют механизмы контроля мета-параметров, влияющих на процедуру обучения. Например, оценка качества работы SVM может вычисляться при различных ядрах.
3. На выходе процедуры тестирования может быть только скалярная величина, что ограничивает возможности детального анализа.
4. Для тестирования алгоритма необходимо иметь его исполняемый код (*java*-класс), что невозможно в случае коммерческих алгоритмов.

**Система MLComp.** Реализуют другую схему работы с алгоритмами – алгоритмы исполняются не на стороне пользователя, а на сервере системы. Для тестирования алгоритма пользователь загружает свою программу на сайт и запускает её выполнение (run) на одной из задач. Программа выполняется в среде Linux. Система не накладывает ограничений на язык программирования. Так как алгоритмы выполняются на одном сервере и имеют одинаковые вычислительные ресурсы, в рамках MLComp можно сравнивать скорости работы алгоритмов. В плане процедур тестирования MLComp значительно уступает TunedIt. Проблема заключается в выбранном подходе к тестированию – формат задачи требует явного указания тестовой и контрольной подвыборок, и эта информация хранится в данных задачи. Все тесты алгоритмов на задаче происходят на одном и том же разбиении, что делает невозможным статистически надежную оценку характеристик качества работы алгоритма.

Ограничения:

1. Только одно разбиение задачи на обучение и контроль, в результате низкая статическая надежность получаемых оценок.
2. Для тестирования алгоритма необходимо передать разработчикам системы его исполняемый код (программу), что невозможно в случае коммерческих алгоритмов.
3. Ограничение на определенную систему (Linux).

Все описанные системы не могут полностью удовлетворить потребности коммерческих организаций для решения задачи выбора алгоритма. Для решения была разработана собственная система, которая лишена недостатков данных решений.

**Собственная система Choice.** Качественно отличается от прошлых систем. Имеет несколько существенных отличий. Главное отличие – архитектура данной системы. Она является распределенной, что позволяет исполнять работу алгоритма на стороне клиента, а все вычисления происходят на стороне сервера **Choice**. Задачи для алгоритмов формируются из запросов пользователей, либо их можно задать самому. Каждый клиент, производящий тестирование, предоставляет часть своих вычислительных мощностей, что делает систему единой, но при этом коммерческая тайна сохраняется, поскольку исходный код не посылается на сервер. За счет этого программа может быть реализована на любом языке программирования, это не повлияет на конечный результат, единственное ограничение – поддержка web-сервисов, что алгоритм мог взаимодействовать с системой. Благодаря этим архитектурным особенностям к системе можно подключать коммерческие версии алгоритмов.

Во-вторых, в **Choice** существенно глубже проработана методика тестирования. Для каждого тестируемого алгоритма можно задать несколько задач, но для каждой будет

использоваться одно множество разбиений, определенное заранее, что позволяет нам обеспечить полную воспроизводимость результатов тестирования, что обеспечит статическую надежность полученных результатов. Помимо стандартного параметра качества: частоты ошибок на контроле – статистика собирается из множества скалярных и графических характеристик для подробного, детального анализа качества алгоритма. Вся информация после тестирования сохраняется на стороне сервера и может выдаваться пользователю сколько угодно раз, в следствии чего мы может полностью выгружать все наши результаты и делать конечный выбор, в зависимости от задачи. **Choice** оперирует с четырьмя типами объектов: задача, разбиение, алгоритм и статистика.

**Репозиторий задач классификации.** Задача классификации задачи *Z* описывается в **Choice** следующим набором данных:

- матрица $F = (Fij) \in R^{L \times n}$ значений j-го признака на i-м объекте;
- вектор $y = (yi)^{L}_{i=1} \in Y^{L}$ правильных ответов;
- матрица $C = (C_{uv}) \in R^{m \times m}$ стоимости потерь при ошибочном соотнесении объекта класса u к классу v ($|Y| = m$);
- вектор информации $I = (I_j)^{n}_{j=1}$ о типах признаков (номинальный, вещественный).

Задачи могут быть загружены в формате APTF либо во внутреннем формате **Choice** в виде нескольких csv-файлов с данными. Пользователь также может предоставить данные своего тестирования общественности либо сделать все приватным.

**Методика тестирования.** Для каждой задачи Z формируется набор разбиений по стандартной схеме кросс-валидации «10x5-fold CV» и девять наборов по десять случайных разбиений с одинаковой длиной обучения (от 10 до 90% от длины выборки с шагом в 10%). В контроль и обучение объекты разных классов попадают в такой же пропорции, как и в исходной задаче. Все разбиения сохраняются, без привязки к алгоритму, а в привязке к задаче. Если задача *Z* однажды тестировалась в **Choice**, то будут использоваться разбиения, что были определены при прошлом тестировании. Этот аспект позволяет нам судить о результатах тестирования в едином контуре. Все алгоритмы тестируются в одинаковых условиях и на одинаковых исходных данных.

**Репозиторий алгоритмов классификации.** Алгоритм классификации *A* принимает на входе:

- матрицу обучающей выборки $F = (F_{ij}) \in R^{l \times n}$,
- вектор $y = (y_i)^{l}_{i=1} \in Y^{l}$ правильных ответов на обучающей выборке,
- матрицу $C = (C_{uv}) \in R^{m \times m}$ стоимости потерь;
- вектор информации I о типах признаков;
- вектор W мета-параметров алгоритма;
- матрицу тестовой выборки $F' = (F'_{ij}) \in R^{k \times n.}$.

И выдает на выходе:

- ответы $y' = (y'_i) \in Y^{k}$ на тестовом наборе;
- матрицу оценок $P' = (P'_{iv}) \in [0,1]^{k \times m}$ принадлежности i-го объекта тестовой выборки v-му классу задачи.

Мета-параметры *W* задаются пользователем перед запуском алгоритма на тестирование. Экземпляры алгоритма *A* с различными значениями мета-параметров *W* (а также различных версий) считаются различными, и результаты их тестирования сохраняются в отдельных записях. Это позволяет **Choice** лучше обеспечивать воспроизводимость по сравнению с TunedIt. Взаимодействие с алгоритмами происходит при помощи специальной прослойки AlgProxy, которую должен реализовать автор алгоритма. AlgProxy обеспечивает обмен данными между алгоритмом *A* и веб-сервисом Полигона. Модель взаимодействия основана на подходе «клиент-сервер», где AlgProxy – это клиент, а **Choice** – это сервер. Инициатором взаимодействия выступает AlgProxy, который постоянно опрашивает **Choice,** «не появились ли задания для алгоритма *A*?». Если ответ положительный, то AlgProxy получает задание и начинает выполнение алгоритма A. Задание содержит данные задачи *Z*,

параметры алгоритма W и набор разбиений S = ($S_q$). AlgProxy тестирует алгоритм *A* на разбиениях *A*, на разбиениях *S* и передает результаты в системе **Choice**. В упрощенном виде схема работы AlgProxy выглядит следующим образом:

- запросить новое задание;
- если задания нет, то через N секунд перейти к первому пункту;
- если задание есть, то запросить данные задачи классификации;
- провести тестирование алгоритма;
- отправить результаты тестирования на сервер Полигона.

**Оценки качества алгоритма для задачи**. Статистика – это функция, которая принимает на входе: данные задачи классификации *Z*; набор разбиений $S = (S_q)$ задачи *Z*, использованных при тестировании алгоритма; результаты работы алгоритма ($y'_q$) и ($P'_q$) на каждом *q*-м разбиении. На выходе функция выдает: скалярное или векторное значение, описывающее одну из характеристик качества алгоритма. **Choice** рассчитывает следующие статистики:

- частота ошибок на обучении и на контроле, а также средняя переобученность (с доверительным интервалом);
- среднее смещение и средняя вариация (характеристики адекватности и устойчивости модели);
- доля «шумовых» (где алгоритм часто ошибается), «пограничных» и «эталонных» (на которых алгоритм ошибается редко) объектов;
- распределение частоты ошибок и переобученности (показатели устойчивости классификации);
- зависимость переобученности и частоты ошибок от длины обучения;
- карта ошибок – точечный график: по оси *X* частота ошибок на обучении, по оси *Y* частота ошибок на контроле, каждая точка на карте соответствует одному разбиению;
- разделение ошибок на смещение и вариацию (анализ качества модели );
- ROC-кривая и площадь под ROC-кривой (анализ соотношения ошибок I и II рода );
- распределение отступов (margin) объектов: по оси *X* – объекты, по оси *Y* – средний отступ объекта от границы класса (ещё одно разделение объектов на «шумовые», «пограничные» и «эталонные»).

Большинство статистик вычисляются как по всей выборке, так и отдельно по классам, а также отдельно для обучающей и контрольной выборки (если это осмысленно). Результаты расчетов могут быть выгружены в «сыром» виде, что позволяет пользователю дополнить имеющиеся статистики собственными.

Таким образом, можно сделать следующие выводы.

Во-первых, для коммерческих проектов реализации систем с машинным обучением крайне важно корректная разработка алгоритма на раннем этапе проекта, а именно его тестирование, поскольку это может привести к огромным издержкам, увеличивающем стоимость проекта в разы.

Во-вторых, тестирование – основа качественной разработки алгоритма машинного обучения. Без него невозможно понять, правильно ли он составлен и решает ли конечную задачу.

В-третьих, для тестирования алгоритма необходимо использовать систему **Choice**. Система не привязана к конкретному языку программирования, система позволяет работать с коммерческими алгоритмами и система предлагает широкий спектр инструментов, позволяющих оттестировать алгоритм по многим параметрам, составить статистики по работоспособности алгоритма и полностью выявить проблемы «обучения» компьютера. Проектирование модели с использование инструмента **Choice** позволит сократить время разработки 1 этапа и позволит выявить ошибки на самом ранней стадии разработки.

**Библиографический список**

1. **Воронцов, К.В.** Система эмпирического измерения качества алгоритмов классификации К.В. Воронцов, А.С. Инякин, С.Ю. Лисица // Всерос. конф. ММРО-13. – М.: МАКС Пресс, 2007. – С. 577-581.
2. **Воронцов, К.В.** «Полигон» - распределённая система для эмпирического анализа задач и алгоритмов классификации / К.В. Воронцов [и др.] // Всеросс. конф. ММРО-14 – М.: МАКС Пресс, 2009. – С. 503-506.
3. **Domingos, P.A.** Unified bias-variance decomposition and its applictions: LCML'17. – 2000. – Pp. 231-238.
4. **Frank, A.** UCI Machine Learning Repository / A. Frank, A. Asuncion // University of California, Irvine, School of Information and Computer Sciences, 2010. www.ics.uci.edu/~mlearn/ MLRepository.html.
5. **Hand, D.** A simple generalization of area under the ROC curve for multiple class classification problems / D. Hand, A. Till // Machine Learning, 45. – 2001. – P. 171-186.

*Дата поступления*
*в редакцию 15.08.2017*

**N.V. Volzhankin, N.V. Zlobina, N.E. Posobilov**

## METHODS FOR TESTING INFORMATION SYSTEMS BASED ON MACHINE LEARNING

Nizhny Novgorod state technical university n.a. R.E. Alekseev

**Purpose:** The goal is to study methods for testing information systems based on machine learning.
**Design/methodology/approach:** For the first stage of the study, it is necessary to conduct a theoretical analysis and generalization of the scientific literature and documents in order to get acquainted with the basic concepts and characteristics of information systems based on machine learning, as well as their testing.
In the second stage, the results of the theoretical analysis are processed, specific methods are developed to support the testing of these systems.
At the third stage, the development of tools to automate the testing process and make it the most effective in terms of time and resources.
**Findings:** To conduct economic analysis of the testing process, to study modern algorithms and criteria for selecting an algorithm for a particular task, to formulate the automation of this choice, to study the data handling module, to develop an automated testing system for the DWH module.
**Research limitations/implications:** The results of the study presented in this paper have limitations in the field of application, since the company is considering companies with powerful infrastructure and a large amount of data. These methods can form the basis for other solutions to solve specific problems.
**Originality/value:** The development of systems allowing to automate the testing process.

*Key words*: machine learning, algorithms, testing, techniques, system design, commercial systems.